THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Gaussbock

Citation for published version:
Moews, B & Zuntz, J 2020, 'Gaussbock: Fast parallel-iterative cosmological parameter estimation with Bayesian nonparametrics', *Astrophysical Journal*, vol. 896, no. 2. https://doi.org/10.3847/1538-4357/ab93cb

Digital Object Identifier (DOI):
10.3847/1538-4357/ab93cb

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Astrophysical Journal

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 23. Jul. 2021

Typeset using LaTeX **twocolumn** style in AASTeX62

arXiv:1905.09800v2 [astro-ph.CO] 16 May 2020

Gaussbock: Fast parallel-iterative cosmological parameter estimation with Bayesian nonparametrics

Ben Moews[1] and Joe Zuntz[1]

[1] *Institute for Astronomy, University of Edinburgh, Royal Observatory, Edinburgh EH9 3HJ, UK*

ABSTRACT

We present and apply Gaussbock, a new embarrassingly parallel iterative algorithm for cosmological parameter estimation designed for an era of cheap parallel computing resources. Gaussbock uses Bayesian nonparametrics and truncated importance sampling to accurately draw samples from posterior distributions with an orders-of-magnitude speed-up in wall time over alternative methods. Contemporary problems in this area often suffer from both increased computational costs due to high-dimensional parameter spaces and consequent excessive time requirements, as well as the need for fine tuning of proposal distributions or sampling parameters. Gaussbock is designed specifically with these issues in mind. We explore and validate the performance and convergence of the algorithm on a fast approximation to the Dark Energy Survey Year 1 (DES Y1) posterior, finding reasonable scaling behavior with the number of parameters. We then test on the full DES Y1 posterior using large-scale supercomputing facilities, and recover reasonable agreement with previous chains, although the algorithm can underestimate the tails of poorly-constrained parameters. Additionally, we discuss and demonstrate how Gaussbock recovers complex posterior shapes very well at lower dimensions, but faces challenges to perform well on such distributions in higher dimensions. In addition, we provide the community with a user-friendly software tool for accelerated cosmological parameter estimation based on the methodology described in this paper.

Keywords: cosmology: cosmological parameters — methods: statistical – methods: data analysis

1. INTRODUCTION

Bayesian methods are now a standard approach to data analysis and inference in astrophysics. In this approach, probabilities are regarded as a means of quantifying information, and in particular the information contained in an experimental dataset about a specific model. This is encoded in the *posterior*, which combines *prior*, or external, information with the *likelihood* from the current data. For textbooks providing an introduction and overview of Bayesian methods, we refer interested readers to Bernardo & Smith (1994), MacKay (2003) , and Gelman et al. (2013), as well as Murphy (2012) and Hobson et al. (2009) for an overview centered on machine learning and cosmology, respectively. In most realistic cases, the analytic or direct numerical evaluation of posterior probability distributions is impossible or infeasible, especially in cases that feature many parameters, due to the large volume of high-dimensional spaces. The wide-spread use of Bayesian methods has largely been driven by the availability of *sampling* algorithms, which can generate samples from a posterior distribution without exploring the full space. These samples can then be used to generate summary statistics like means and limits on individual parameters, or correlations between them. For a shorter overview of the application of Bayesian inference in cosmology, see Trotta (2008).

Within the cosmology literature, Christensen et al. (2001) proposed initial arguments for the use of Bayesian methods for the purpose of cosmological parameter estimation. They argued for *Markov chain Monte Carlo* (MCMC) approaches due to their superiority in terms of sampling from, and converging to, the true posterior distribution in the limit of an infinite sample size. The application of MCMC approaches in these early efforts were centered on the *Metropolis-Hastings algorithm*, which was named after work done by Metropolis et al. (1953) and, for the more general case, Hastings (1970). The distinguishing feature of this method is the acceptance of new points in the Markov chain if the likelihood ratio of the proposed point and the last point is larger

Corresponding author: Ben Moews
bmoews@roe.ac.uk

than one, and the probabilistic acceptance of points with a lower ratio if the latter is larger than a random number $n \in [0, 1]$. This acceptance of less likely points dependent on the likelihoods leads to the sampling from the posterior distribution and, notably, does not require marginalization via the evidence. Knox et al. (2001) then followed the proposal of Christensen et al. (2001) to constrain the age of the universe to $t_0 = 14.0 \pm 0.5$ Gyr. Earlier work includes Saha & Williams (1994), who made use of the Metropolis-Hastings algorithm for for galaxy kinematics, and Christensen & Meyer (1998), who employed the related Gibbs sampler for gravitational wave analysis (Geman & Geman 1984). For more in-depth information covering the wide array of contributions from both the astrophysical and statistical literature, we recommend Trotta (2008) as a more complete overview of the development of Bayesian inference in cosmology in particular, and Robert & Casella (2011) and Brooks et al. (2011) for a history of MCMC methods and their development in general.

Up to, and into, the new millennium, the Metropolis-Hastings algorithm remains the standard approach to cosmological parameter estimation, which was further supported by the development of a dedicated implementation in `CosmoMC` (Lewis & Bridle 2002). A variety of algorithms and codes are, however, available for different types of problems. The optimal choice depends on multiple factors, including the dimensionality of the problem, meaning the number of parameters to estimate, the evaluation speed, the need for Bayesian evidences, the availability of analytic derivatives, the ability to sample from marginal distributions, and the possibility and degree of using parallelization.

In more recent years, new MCMC sampling techniques were proposed and subsequently applied to cosmological parameter estimation. Examples include *Population Monte Carlo* (PMC) techniques introduced by Cappé et al. (2004) and Wraith et al. (2009), and used by Kilbinger et al. (2010) to develop `CosmoPMC`; *affine-invariant MCMC ensembles* by Goodman & Weare (2010), which led to the publication of `emcee` by Foreman-Mackey et al. (2013) and `CosmoHammer` by Akeret et al. (2013); and renewed interest in *Approximate Bayesian Computation* (ABC) for likelihood-free inference based on simulations to introduce `CosmoABC` and `abcpmc` (Ishida et al. 2015; Akeret et al. 2015). *Density estimation likelihood-free inference* (DELFI) is a recently developed technique that trains a flexible density estimator to approximate the target posterior, circumventing the large number of simulations that traditional ABC approaches can require (Bonassi et al. 2011; Fan et al. 2013; Papamakarios & Murray 2016). Using the JLA sample of 740 type SN Ia supernovae as described in Betoule et al. (2014), Alsing et al. (2018) subsequently deploy this method to estimate cosmological parameters. Their approach, however, makes a few simplifying assumptions, for example normally distributed priors and likelihoods. Other advanced methods, like the *Hamiltonian Monte Carlo* approach developed by Duane et al. (1987), have also been applied, for example by Hajian (2007). These developments are driven by the computationally costly likelihood calculations involved in most MCMC algorithms, trying to alleviate this issue with a certain degree of parallelization due to the increased availability of cheap computing resources, faster convergence or, in the case of ABC, the circumvention of direct likelihood computations altogether.

As such methods either fail to reduce the runtime enough for modern problems or have their own pitfalls, for example through an increased risk of introducing biases, the quest for highly parallelized and fast alternatives for cosmological parameter estimation continues. This need is further exacerbated by upcoming missions like LSST and Euclid requiring high-dimensional posterior approximations with a large number of required nuisance parameters predicted to vastly exceed previous missions (Amendola et al. 2018).

It should be noted that the statistical literature on sampling methods is rich and vast, and a complete review of both their history and all current developments would exceed the scope of this paper. The methods covered in more detail here are those likely to be more familiar to the astrophysical community, due to being wide-spread or featuring field-specific implementations. While we aim to cover relevant comparisons, this should, of course, not be misunderstood as a judgment about these methods being superior in the wider context of all statistical developments, but to place this work in the context of astrostatistics.

Nested sampling is a Bayesian take on numerical Lebesque integration for model selection introduced by Skilling (2006). While targeting the calculation of Bayesian evidence, posterior samples are generated as a by-product, and the algorithm was quickly shown to require considerably fewer posterior evaluations (Mukherjee et al. 2006). Due to denser and sparser sampling from high-posterior and low-posterior regions, respectively, nested sampling provides increased efficiency when compared to previous MCMC methods. This has lead to extensions and implementations for applications in cosmology, notably `CosmoNest` by Liddle et al. (2006), `MultiNest` as described in Hobson & Feroz (2008) and Feroz et al. (2009), and

`PolyChord` (Handley et al. 2015). In cosmology, such implementations have been used in areas as diverse as cosmic ray propagation models, cosmoparticle physics, and gravitational wave astronomy (Trotta et al. 2011; Del Pozzo 2012; Verde et al. 2013; Del Pozzo et al. 2017; Wang et al. 2018). A comparison between nested sampling and state-of-the-art MCMC methods can be found in Allison & Dunkley (2014), while an investigation of statistical uncertainties in nested sampling is provided by Keeton (2011). Nested sampling has also been adopted by other fields of research, including GPU-accelerated implementations, for example in systems biology (Aitken & Akman 2013; Stumpf et al. 2014).

The statistical literature, however, points out various issues of nested sampling methods that have prevented wide-spread adoption in statistics. Among these are the assumption that perfect and independent samples from a constrained version of the prior are drawn in each iteration, the underestimation of sampling errors due to the simulated-weights method it employs, and an asymptotic approximation variance that scales linearly with the dimensionality of a given parameter space (Chopin & Robert 2010; Higson et al. 2018).

In this paper, we use example likelihoods from the Dark Energy Survey (DES) collaboration's analysis of lensing and clustering data, as presented in Abbott et al. (2018). These calculations make use of the `CosmoSIS` and `CosmoLike` pipelines, which contain implementations of both `Multinest` and `emcee` (Zuntz et al. 2015; Krause & Eifler 2017).

For a comparison of approaches designed for the acceleration of MCMC methods in particular, including additional parallelization methods, see Robert et al. (2018), who cover methods targeting both the exploration stage of the algorithms and the exploitation level. The second approach includes Rao-Blackwellization and scalability, with the latter encompassing parallelization under this nomenclature. Other examples of methods trying to optimize the performance of established algorithms include the *no-U-turn sampler* (NUTS) by Hoffmann & Gelman (2014), which alleviates the need of the previously mentioned HMC algorithm for tuning by computing the trajectory length via recursively built candidate proposals, as well as work by Neiswanger et al. (2014) on asymptotically exact and embarrassingly parallel MCMC sampling. The latter solves the slowing-down of parallel MCMC methods by reducing the amount of required communication in a divide-and-conquer tactic that splits up the dataset and which the authors justify with prohibitively long runtimes of many serial methods. Interestingly, our method allows for the use of any sampling method to create the initial sample, meaning that such optimized divide-and-conquer methods can be easily incorporated into our approach. The need for sped-up posterior estimation approaches is further elaborated on by Bardenet et al. (2014) and Wilkinson (2005), with the latter pointing out the need for parallelized method due to: "[...] weeks of CPU time on powerful computers" for serial MCMC methods on high-dimensional problems of interest. The need for parallelization approaches stems mostly from cases in which parts of the computations are very expensive, but which can be transformed into an, ideally, embarrassingly parallel problem that allows the respective steps to take full advantage of a greater number of cores, thus cutting otherwise infeasible runtimes to a fraction. For a more general overview of both the history and more recent developments in the field of Monte Carlo methods, such as multi-stage Gibbs samplers, see Robert & Casella (2004).

In this paper, we propose a parallel-iterative algorithm to address these issues, making use of recent advances in the fields of statistics and machine learning. Our method starts with a preliminary approximation of the target distribution, either through a built-in affine-invariant MCMC ensemble or a user-provided initial sample guess. We then fit a non-parametric model to the sample and employ a variation of sampling-importance-resampling to iteratively move the samples toward the true distribution, repeating these steps until the process converges. In doing so, we also offer a user-friendly Python package to both the cosmology and the wider astronomy community, as well as a general parameter estimation tool for other disciplines dealing with the same issues. We test our implementation on the DES Year 1 (Y1) posterior, and on a fast approximation to the latter for extended tests.

The remainder of this paper is structured as follows: We cover the relevant methodology, which includes an overview of variational inference for Bayesian mixture models and truncated importance sampling, as well as the mathematical architecture of the proposed approach, in Section 2. In Section 4, we introduce an open-source implementation based on our method, explain computational considerations and parallelization, and provide a quickstart tutorial. Experiments for both toy examples and an approximation of the DES Y1 likelihood are covered in Section 5, together with cosmological parameter estimation runs on supercomputing facilities for the full DES Y1 likelihood. We present and discuss the results of these experiments in Section 6, and summarize our findings in the conclusion in Section 7.

2. MATHEMATICAL BACKGROUND

While Bayesian inference has earned its place as a powerful instrument for cosmological research, complex problems often suffer from the need to approximate probability densities that are difficult or outright infeasible to compute. Since Bayesian methods rely on the posterior density, approximations are a necessary evil. In the algorithm presented in this paper, we fit a mixture model to sample from the posterior using *variational inference* methods, while avoiding fixing the number of mixture components by using a *Dirichlet process*. We iteratively improve these samples using *truncated importance sampling* until a convergence criterion is fulfilled.

In this section, we introduce variational inference in Section 2.1, followed by Dirichlet processes and the stick-breaking procedure in Section 2.2. After an overview of sampling-importance-resampling and truncated importance sampling in Section 2.4, we introduce a novel method for parallel-iterative parameter estimation in Section 3.

2.1. *Variational inference for mixture models*

Variational Bayesian methods were originally developed and explored in the context of artificial neural networks, and gained initial interest from research on inference in graphical models (Peterson & Anderson 1987; Peterson & Hartman 1989; Jordan et al. 1999). The use of variational Bayesian methods for inference is commonly known as *variational inference* (VI) and provides a faster and more scalable alternative to Markov chain Monte Carlo (MCMC) methods in many contexts; the main difference between them is that VI treats parameter estimation not as a sampling problem, but instead as an optimization problem. From a research point of view, these methods also garnered the interest of the statistics community because they are currently not as well understood as MCMC methods (Blei et al. 2017).

The *Kullback-Leibler divergence* D_{KL} is a central concept in VI and defines by how much a distribution diverges from another, or how similar it is. For a reference distribution $p(\mathbf{x})$ and a proposal distribution $q(\mathbf{x})$, the D_{KL} can be expressed as

$$D_{\mathrm{KL}}(p(\mathbf{x})||q(\mathbf{x})) = \int_{-\infty}^{\infty} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}\, d\mathbf{x}. \tag{1}$$

The fact that the D_{KL} is an asymmetric difference measure means that $D_{\mathrm{KL}}(p(\mathbf{x})||q(\mathbf{x})) \neq D_{\mathrm{KL}}(q(\mathbf{x})||p(\mathbf{x}))$, which is due to its calculation as a directional loss of information.

In VI, the D_{KL} is used to find a best-fitting distribution to a set of samples. Let $\mathscr{Q}$ be a selected family of distributions, $\mathbf{x}$ and $\mathbf{z}$ observations and parameters, respectively, and $p(\mathbf{z})$ a prior density that can be related to observations via the likelihood $p(\mathbf{x}|\mathbf{z})$ to calculate the posterior $p(\mathbf{z}|\mathbf{x})$. The family member $\hat{q}(\mathbf{z})$ that best matches the posterior can be found in the framework of an optimization problem, finding with some specified tolerance the value of

$$\hat{q}(\mathbf{z}) = \underset{q(\mathbf{z})\in\mathscr{Q}}{\operatorname{argmin}}\, D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})\right). \tag{2}$$

Calculating this quantity directly is often infeasible, since it is equivalent to measuring the Bayesian evidence. Instead, VI methods (equivalently) maximize an alternative quantity, the *evidence lower bound* (ELBO),

$$\begin{aligned} \mathrm{ELBO}(q) &= \mathbb{E}[\log\, p(\mathbf{z},\mathbf{x})] - \mathbb{E}[\log\, q(\mathbf{z})] \\ &= \mathbb{E}[\log\, p(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}\left(q(\mathbf{z})||p(\mathbf{z})\right), \end{aligned} \tag{3}$$

which is numerically easier to calculate than the D_{KL}. The ELBO also delivers a lower bound for the evidence, which is the reason for the utility of VI for model selection, as covered in Blei et al. (2017). A more extensive introduction to VI for the interested reader can be found in Murphy (2012).

2.2. *Dirichlet processes and stick-breaking*

Instead of the more traditional approach of fixing the number of components in the mixture model that we use to model the posterior, we determine the component number from the sample itself at each iteration. This approach employs a *Dirichlet Process* (DP) as a prior on the number of parameters, which enables the use of a suitable number of components during each step, meaning that changes between iterations are not forced to use the same components.

Developed by Ferguson (1973), DPs are distributions of distributions, featuring a base distribution G_0 and a scaling parameter $\alpha \in \mathbb{R}_+$, and with realizations denoted as $G \sim \mathrm{DP}(\alpha, G_0)$. This area has important applications as the prior in infinite mixture models, and gained new traction in both statistics and machine learning in recent years (Gershman & Blei 2012). The DP mixture model presented originally by Antoniak (1974) takes θ_i as the distribution parameter of observation i and uses the discrete nature of the base distribution G_0 to view the DP mixture as an infinite mixture model. For samples $\mathbf{s}$ from such a DP mixture, with sample size N, the predictive density with $\mathbf{s} = s_1, ..., s_N$ is

$$p(s|\mathbf{s}, \alpha, G_0) = \int p(s|\theta) p(\theta|\mathbf{s}, \alpha, G_0)\, d\theta. \tag{4}$$

As the computation of that density is, again, infeasible, Blei & Jordan (2006) introduce the use of VI for DP mixtures. Bayesian takes on mixture models employ a

prior over the mixing distribution as well as over the cluster parameters, with the former commonly being a Dirichlet and the latter being a Gaussian distribution in our case. Given the discrete nature of random measures drawn from a DP, a mixutre of the latter can be viewed as a mixture model with an unbounded number of components (Blei & Jordan 2006).

The Bayesian nonparametrics approach employs the *stick-breaking process* by Sethuraman (1994), which exploits the discrete nature of DPs to calculate the probability mass function, and can be used for Bayesian Gaussian mixtures with an undetermined number of Gaussians. The name is based on the analogy of breaking a stick of unit length into infinite segments by consecutively breaking off β_1, β_2, etc. from the stick until the remainder is truncated to recover a finite-dimensional representation. The truncated variational distribution is then used to approximate the posterior of an infinite DP mixture. As a mathematical description of the subsequent application of VI to DPs with stick-breaking would go beyond the scope of this overview, we refer the reader to Blei & Jordan (2006). A less concise introduction to DPs and Bayesian nonparametrics in general, as well as its applications, is provided in Hjort et al. (2010).

As the posterior distribution, given a DP mixture prior, cannot be directly calculated, VI offers a deterministic approach to approximate them. In this paper, we employ the mean-field family within VI to optimize the D_{KL}, using this approach to approximate the joint posterior for parameters of an infinite Gaussian mixture, made finite to a maximum number of components through stick-breaking.

2.3. *Importance Sampling*

Importance sampling was described early by Kahn & Marshall (1953) in the context of sample size reduction in Monte Carlo methods and continues to inspire a wide array of extensions. This includes physics-specific techniques like umbrella sampling for difficult energy landscapes by Torrie & Valleau (1977) and, more recently, methods to alleviate issues with poorly approximated proposal distributions (Ionides 2008). It is also a staple in cosmological parameter estimation, for example in Wraith et al. (2009) and Kilbinger et al. (2010). Generally, the basic method is a way to estimate distribution properties if only samples from a different, often approximated, distribution are given. Let $p(\mathbf{z})$ be the target distribution, $q(\mathbf{z})$ an approximate (or proposed) distribution, and $f(\mathbf{z})$ some function. The expectation of $f(\mathbf{z})$ can then be computed as

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z})\ d\mathbf{z} \\ &= \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})\ d\mathbf{z} \\ &\simeq \frac{1}{N}\sum_{i=1}^{N}\frac{p(\mathbf{z}_i)}{q(\mathbf{z}_i)}f(\mathbf{z}_i),\end{aligned} \tag{5}$$

with N as the number of drawn samples. The ratios in this equation, given as

$$r_l \equiv \frac{p(\mathbf{z}_l)}{q(\mathbf{z}_l)}, \tag{6}$$

are called the *importance weights* or *importance ratios* and are central to the method.

Sampling-importance-resampling (SIR) is a two-step approach in which the importance weights for a set of samples are calculated, after which an equally-sized subset of these samples is generated by drawing from them with probabilities per sample indicated by the normalized importance weights. For a more in-depth introduction to importance sampling and other related sampling methods, see Bishop (2006).

2.4. *Counteracting high-weight samples*

One issue with this approach is the possibility of overly dominant samples, meaning points with disproportionately high posterior values in comparison to the rest of a set of model samples. During the importance resampling step, this dominance leads to copies of these samples being overrepresented, resulting in sets that are too narrow in their densities. We address this issue with *truncated* importance sampling, an extension of importance sampling that truncates weights of high-value samples based on the total number of drawn samples, with guarantees for finite variance and mean-square consistency under weak conditions (Ionides 2008). For a set of N_i samples, proposal distribution posteriors $q(\theta_i)$, actual posteriors $p(\theta_i)$ and a set truncation value α with justifications to be set at $\alpha = 2$, the weight w_i of a single sample is updated according to

$$w_i = \min\left(r_i, \bar{r}N_i^{\frac{1}{\alpha}}\right), \text{with } r_i = \frac{p(\theta_i)}{q(\theta_i)}, \tag{7}$$

where $\bar{r}$ is the mean of all importance weights for the sample. With this extension applied to SIR, the weighted drawing of samples is limited by the truncation value. This change improves the behavior of importance sampling during the early part of the algorithm described below, when the estimated distribution q is a poor approximation to the desired posterior p, and alleviates the issue of working with relatively small sample sizes for high-dimensional parameter spaces.

3. THE GAUSSBOCK ALGORITHM

Based on Bayesian nonparametrics and machine learning as described in Sections 2.1 and 2.2, we introduce an algorithm that uses variational inference on an infinite Dirichlet process approximated via stick-breaking to fit variational Bayesian Gaussian mixture models (GMMs) in an iterative manner. This algorithm offers a highly adaptive and embarrassingly parallel way to approximate high-dimensional posteriors with computationally expensive likelihoods.

Data: Initial posterior-space samples $\boldsymbol{\theta}_{\text{start}}$,
number of required output samples n,
array of allowed ranges per parameter $\mathbf{r}$,
number of samples drawn per iteration m,
safety margin multiplier for sampling c,
maximum number of mixture components g,
dynamically shrinking fitting tolerance d,
value for importance weight truncation α,
log-posterior function for $p(\boldsymbol{\theta}|\mathcal{D})$
Result: Approximated posterior samples $\boldsymbol{\theta}_{\text{final}}$
$\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{start}}$;
for $i \leftarrow 1$ ***to*** N **do**
  Calculate the (shrinking) model fitting tolerance
  $d \leftarrow a_1 - i \cdot \Delta a \cdot (N-1)^{-1}$
  Fit a variational Bayesian GMM to the samples
  $\mathcal{M}_i \leftarrow \text{VBGMM}(\boldsymbol{\theta}_{\text{new}}, d, g)$
  Sample a set of parameters from the fitted model
  $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta} \sim \mathcal{M}_i$ s.t. $\text{length}(\boldsymbol{\theta}) = m \cdot c$
  Cut samples straying beyond the allowed ranges
  $R = \text{r}_1 \times \text{r}_2 \times \ldots \times \text{r}_{\dim(\boldsymbol{\theta}_i)}$
  $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i \cap R$
  Keep the required number of parameter samples
  $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i^{(1:n)}$
  Parallel calculation of true log-posterior values
  $\mathbf{p} \leftarrow p(\boldsymbol{\theta}_i|\mathcal{D})$
  Compute importance probabilities in linear space
  $\mathbf{w}_i \leftarrow \exp(\mathbf{p} - p(\boldsymbol{\theta}_i|\mathcal{M}))$;
  Compute the truncated importance probabilities
  $\mathbf{w}_i \leftarrow \min(\mathbf{w}_i, \bar{\mathbf{w}}_i \cdot \text{length}(\boldsymbol{\theta}_i^{\frac{1}{\alpha}}))$
  Renormalize the updated importance probabilities
  $\mathbf{w}_i \leftarrow \mathbf{w}_i \times (\sum \mathbf{w}_i)^{-1}$;
  Weighted sampling from the parameter samples
  $L \leftarrow \text{length}(\boldsymbol{\theta}_i)$
  $\boldsymbol{\theta}_{\text{new}} \leftarrow \text{sample}(\boldsymbol{\theta}_i, \mathbf{w}_i)$ s.t. $\text{length}(\boldsymbol{\theta}_{\text{new}}) = L$
  Terminate if convergence criterion is reached
  if $|\Delta\sigma_i^2 < t$ **then**
    break
  end
end
Return the user-specified number of final samples
return $\boldsymbol{\theta}_{\text{final}} \leftarrow \boldsymbol{\theta} \sim \mathcal{M}_i$ s.t. $\text{length}(\boldsymbol{\theta}) = n$

Algorithm 1: Pseudo-code for `Gaussbock`.

The idea behind our approach is to start from an initial sample guess, either from existing data or a short run of another sampler such as `emcee`. Based on the work on nonparametric VI by Gershman & Blei (2012), our algorithm uses a variational Bayesian GMM due to its ability to automatically determine the number of Gaussians required to produce a good fit by stick-breaking an infinite Dirichlet process mixture. For this reason, only the provision of a maximal number of Gaussians is required. The algorithm then determines means and variances for the optimal number of Gaussians given a sample and fitting tolerance. This is followed by drawing a new sample from the fitted mixture model, and a truncated SIR step to move the sample distribution further toward the true the posterior density. These steps are then repeated in an iterative manner until convergence, which is assessed from the change in the variance of importance weights at the end of each iteration:

1. Fit a variational Bayesian GMM to the sample,
2. draw a new sample from the newly fitted model,
3. perform an SIR step for a weighted sample, and
4. check inter-iteration variances for convergence.

We use a dynamically shrinking tolerance d for the model-fitting step. Let a be the tuple denoting the initial and final model-fitting tolerances, with $a_1 > a_2$, and let N be the maximum number of iterations, then the tolerance d_i for a given iteration $i \in \{1, 2, \ldots, N\}$ is

$$d_i = a_1 - i \cdot \Delta a \cdot (N-1)^{-1}, \text{ with } \Delta a = a_1 - a_2. \quad (8)$$

This approach is related to the previously mentioned PMC algorithms initially introduced by Cappé et al. (2004), and applied to cosmological inference in Kilbinger et al. (2010). It differs, though, by the nonparametric nature of the model, which eliminates the bias present in the predetermined number of distributions in classical GMMs. It also adds the weight truncation to reduce the influence of overly dominant samples with high posterior values in relatively small samples. Our method bears motivational similarity, although considerable methodological differences, to `CosmoABC`, while not being subject to the potential pitfalls of forward-simulation inference in ABC (Ishida et al. 2015).

In Algorithm 1, we provide a more complete pseudo-code representation of the most relevant parts of the approach described in this paper, which we name `Gaussbock`. For this algorithm, we let $\mathbf{r}$ be the array of tuples representing the allowed ranges (min, max) per dimension, that is, per parameter. Furthermore, let N be the maximum number of iterations, m the number of samples to be drawn from each iteration's model, n the number of samples returned after termination,

g the maximum number of Gaussians available for approximating the posterior distribution, and c a safety margin parameter greater than one to draw additional GMM samples in case some fall outside the parameter bounds. Finally, let $\mathcal{D}$ be the empirical data used for calculating the true likelihood. The specifics of the variational Bayesian GMM (VBGMM) with reasonable default settings, like the prior of the covariance distribution and the parameter initialization for the VBGMM, are omitted in order to keep the pseudo-code concise.

As (mostly adaptive) defaults are used for the settings of `Gaussbock`, only the initial approximative sample set $\boldsymbol{\theta}_{\text{start}}$, the number of iterations n, and the handle of a function to compute $p(\boldsymbol{\theta}_i|\mathcal{D})$ have to be provided with regard to the above pseudo-code. In addition, if no $\boldsymbol{\theta}_{\text{start}}$ is provided, the implementation described in Section 4 will automatically run an affine-invariant MCMC ensemble to procure that initial set of posterior-space samples. Since the determination of convergence is a common issue in MCMC methods, `Gaussbock` uses a convergence threshold t that terminates the iterative fitting-resampling procedure if reached before the maximum number of iterations. For this purpose, we measure the difference in inter-iteration weight variances $\Delta\sigma_i^2$, which takes the form

$$\begin{aligned} &\Delta\sigma_i^2 = |\bar{\sigma}_i^2 - \bar{\sigma}_{i-1}^2|, \\ &\text{with } \bar{\sigma}_i^2 = \dim(D)^{-1} \sum_{d=1}^{\dim(D)} \sigma(\log(w_{i_d}))^2. \end{aligned} \tag{9}$$

Here, the average logarithmic importance weight variance is denoted as $\bar{\sigma}_i^2$, providing the arithmetic mean over the dimensionality $\dim(D)$, meaning the number of parameters.

4. SOFTWARE IMPLEMENTATION

In order to make this algorithm readily available, we have released a Python 3 package incorporating the complete `Gaussbock` algorithm. The package is installable via `pip` from the Python Package Index[1], while documentation and source code are available in a public repository[2].

Figure 1 shows the schematic workflow of `Gaussbock`, with a choice between an automated initial posterior approximation and a user-provided sample guess, as well as the option to return importance weights and the final fitted model. The automated initial approximation makes use of an affine-invariant MCMC ensemble, as introduced by Goodman & Weare (2010), through the package `emcee` by Foreman-Mackey et al. (2013) and with parameters like the number of walkers being automatically determined based on the required function inputs. The only required inputs to the tool's main function are the lower and upper limits for each parameter ('parameter_ranges'), the handle of a function that accepts a point in the problem's parameter space and returns its log-posterior value ('posterior_evaluation'), and the desired number of posterior samples to be returned ('output_samples').

An overview of settable inputs is shown in Table 1. We strongly encourage users to provide parameter ranges that are scaled to the interval $[0, 1]$ when setting a threshold for the optional convergence determination ('convergence_threshold') due to its mean variance-

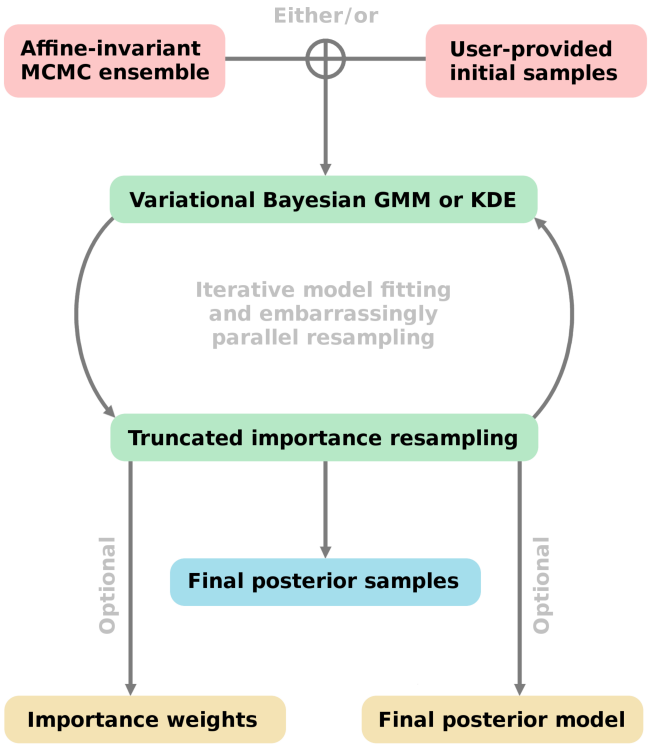

Figure 1. Schematic workflow of `Gaussbock`. Inputs are colored in red, iterative steps in green, primary outputs in blue, and optional outputs in yellow. Starting with an initial et of samples that roughly appproximates the posterior distribution, the method uses an iterative model-fitting and parallelized sampling-importance-resampling step using importance ratio truncation to evolve toward tighter fits for the true posterior. Depending on the dimensionality of the problem, a variational Bayesian Gaussian mixture model (GMM) or kernel density estimation (KDE) can be used. This iterative step is is repeated until convergence or a maximum number of iterations is reached. As indicated by the exclusive OR connection, the initial sample set can be user-provided or automatically inferred through a short-chained affine-invariant Markov chain Monte Carlo (MCMC) ensemble.

[1] https://pypi.org
[2] https://github.com/moews/gaussbock

Table 1. `Gaussbock` inputs. The table lists all 19 possible inputs that can be set by the user, as well as a short explanation for each input, with the first three being required. The remaining 16 optional inputs are marked with an asterisk before their name and are default values are based on the tests presented in this paper and should, as a result, generally achieve desirable performance for a wide array of problems reasonably similar to those described here.

Input	Explanation	Default
1. `parameter_ranges`	The lower and upper limit for each parameter	
2. `posterior_evaluation`	Evaluation function handle for the posterior	
3. `output_samples`	Number of posterior samples that are required	
4. `*initial_samples`	Choice of `emcee` or a provided start sample	['automatic', $2 \cdot \dim(D) + 2$, 10^3]
5. `*gaussbock_iterations`	Maximum number of Gaussbock iterations	10
6. `*convergence_threshold`	Threshold for inter-iteration convergence checks	None
7. `*mixture_samples`	Number of samples drawn for importance sampling	10^4
8. `*em_iterations`	Maximum number of EM iterations for the mixture	10^3
9. `*tolerance_range`	The range for the shrinking convergence threshold	$[10^{-2}, 10^{-7}]$
10. `*model_components`	Maximum number of Gaussians fitted to samples	$\mathrm{ceil}((2/3) \cdot \dim(D))$
11. `*model_covariance`	Type of covariance for the GMM fitting process	'full'
12. `*parameter_init`	How to intialize model weights, means and covariances	'random'
13. `*model_verbosity`	The amount of information printed during runtime	1
14. `*mpi_parallelization`	Whether to parallelize Gaussbock using an MPI pool	False
15. `*processes`	Number of processes Gaussbock should parallelize over	1
16. `*weights_and_model`	Whether to return importance weights and the model	False
17. `*truncation_alpha`	Truncation value for importance ratio reweighting	2.0
18. `*model_selection`	Type of model used for the fitting process	'gmm' if $\dim(D) > 2$, else 'kde'
19. `*kde_bandwidth`	Kernel bandwidth used when fitting via KDE	0.5

based functionality. When setting a convergence threshold, we recommend a value of $\sim 0.01 \cdot \dim(D)$ as a choice that, based on the tests performed in the course of this work, takes increased dimensionalities into account when using the built-in convergence criterion. The implementation uses `schwimmbad`, a library for parallel processing tools, to provide MPI parallelization on parallel computing architectures (Price-Whelan & Foreman-Mackey 2017). The use of MPI can be activated with the optional boolean input ('mpi_parallelization') being set to 'True'. Alternatively, for running the algorithms across multiple cores locally, the optional input 'processes' can be set to the number of desired cores to be used. The initial sample to start from can be provided by the user, for example through sampling a best-guess approximation or using the posterior from previous research ('initial_samples').

An input of special importance is the ability to set the variable parameter for truncated importance sampling ('truncation_alpha'), the ideal value of which can change based on the difficulty of the posterior approximation problem. By default, the recommended value of 2.0 is used (Ionides 2008). When dealing with, for example, high-dimensional truncated Gaussians or similarly hard-to-approximate shapes, a value of up to 3.0 can enforce a stronger truncation to combat high-weight samples. Similarly, the truncation value can be set down to a minimum of 1.0 for weaker importance weight truncation. Interlinked with this input are the dimensionality of the problem and number of samples drawn from a fitted model in each iteration ('mixture_samples'), as a lower number of samples in a higher-dimensional parameter space increases the odds of importance weights with comparatively high values due to sparse samples. Time requirements and the number of available cores are the limiting factors for such considerations, which is discussed in the experiments in Section 5.

The algorithm's runtime can be further influenced by limiting the maximum number of Gaussians to be used for fitting a VBGMM during each iteration ('model_components'). By default, this input is determined based on the number of parameters to be estimated, but user knowledge about the complexity of the target distribution can inform the requirement for lower or higher maximums. Low-dimensional problems with $\dim(D) < 3$ trigger the use of kernel density estimation (KDE) instead of a VBGMM by default, as this density estimation approach is quite powerful in such scenarios, but faces issues in higher-dimensional problems (O'Brien et al. 2016). The use of KDE or a VBGMM can, however, be forced by the user by setting the respective optional input ('kde_bandwidth') to

either 'kde' or 'gmm'. The bandwidth used for the KDE functionality can be customized with an optional input ('kde_bandwidth'). We advise the use of KDE for low-dimensional problems due to the ability to catch hard-to-approximate posteriors in combination with our iterative method, which we demonstrate in Section 5.4.

5. EXPERIMENTS

DES is an imaging survey that covers 5000 square degrees of the southern celestial hemisphere, operating a wide-field camera on the 4-meter Vctor M. Blanco Telescope located at the Cerro Tololo Inter-American Observatory (Abbott et al. 2016a). The survey probes cosmology using multiple different sources, including galaxy clustering and lensing, cluster counts, and supernova measurements. Preliminary constraints from DES Science Verification (SV) data are presented in Abbott et al. (2016b) and Kacprzak et al. (2016) while, more recently, results and data for DES Y1 observations are described by Abbott et al. (2018) and have been made public[3].

In this work, we use the Y1 weak lensing and galaxy clustering measurements as a test of `Gaussbock`. These measurements consist of a set of 2D two-point correlation functions of galaxy shape and position ("3x2pt") in tomographic bins by redshift. These functions can be predicted from the cosmological matter power spectrum and redshift-distance relation, both of which are sensitive to the underlying cosmological parameters, and especially to the matter density fraction Ω_m and the variance of cosmic structure σ_8. DES analyses yield constraints on these parameters comparable to those obtained from the CMB with Planck (Aghanim et al. 2018). For our experiments, we use the baseline ΛCDM model with varied neutrino density as our test likelihood. The sampling methods used in the main DES analysis are discussed in Krause et al. (2017); they use both the `emcee` affine-invariant sampler and the `MultiNest` nested sampling method, and found close agreement between the two methods.

In Section 5.1, we describe a fast-likelihood approximation of the DES Y1 posterior, followed by a performance test for `Gaussbock`. We explore scaling behavior of our implementation on the same approximation with experiments in Section 5.2. In Section 5.3, we run `Gaussbock` on the full DES Y1 posterior to test both the performance in real scenarios and the the ability to run fully parallelized via MPI on supercomputing facilities. Lastly, in Section 5.4 we test the behavior of the method on distributions with specific challenges and determine what types of failure modes it experiences.

[3] https://des.ncsa.illinois.edu/releases/y1a1

5.1. *Approximating the Dark Energy Survey posterior*

The real DES Y1 likelihood is slow to evaluate, with durations per likelihood that make serial algorithms non-viable, as in Wilkinson (2005). In order to enable experiments that target controlled assessment and scaling behavior, we use an approximation to the DES Y1 posterior with a multivariate truncated Gaussian distribution, for which we employ the mean and covariance values for 26 cosmological and nuisance parameters, as well as their limits from the respective DES data release. This approach results in an extremely fast parameter set evaluation based on a DES Y1 approximation suitable for our purposes. A perfectly Gaussian approximation to the posterior would be an artificially easy test of a model that fits Gaussians; our posterior is truncated within a few sigma of the peak in many of its parameters, and thus provides a reasonable challenge.

As discussed in Section 4, we use an increased truncation value for the SIR step of `Gaussbock`, which we set to 3.0, and a convergence threshold of $0.01 \cdot 26 = 0.26$ that follows the previously outlined best-practice guidelines and triggers the use of the built-in convergence determination. The number of samples per iteration is set to 15000, with the reasoning behind this choice further outlined in Section 5.2. As we want to weight the returned posterior samples with their importance weight, we activate the additional return of the final model and importance weights. Apart from these settings, we use the default behavior of `Gaussbock` by not providing other optional inputs. Table 2 shows the lower and upper limits for cosmological and nuisance parameters employed in our approximation.

The results of this experiment are shown in Figure 2 and demonstrate the abilily of `Gaussbock` to recover correct constraints. Starting from a short and unconverged `emcee` chain, for which distributions are shown in yellow, the importance-weighted posterior samples marked in blue closely match the long-run `emcee` samples highlighted in red. The achieved level of agreement is good enough to make posterior contours and distributions for the target distribution and the importance-weighted samples hard to separate by eye. While the distributions for unweighted posterior samples in green show a good agreement with the long-run samples, weighting the output samples with the optionally returned importance weights pushes the sample distributions further toward to target posterior, thus validating the additionally provided functionality related to KDE for low-dimensional parameter estimation. While this experiment is based

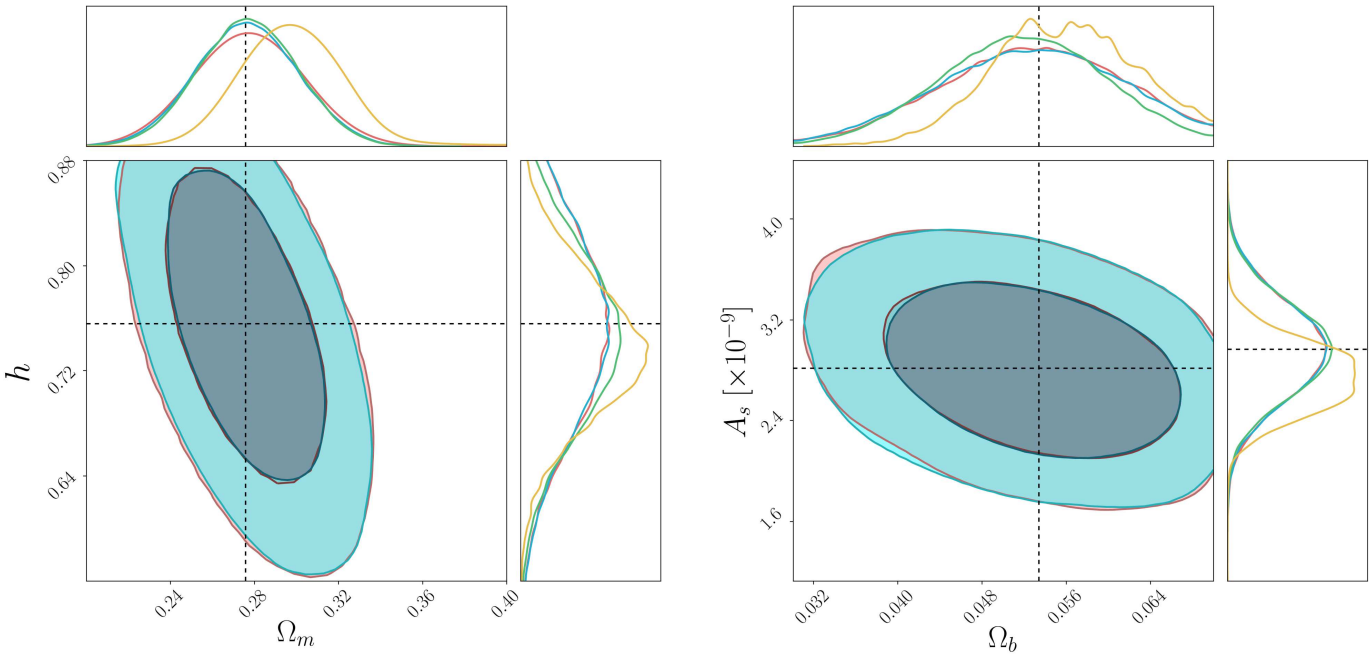

Figure 2. DES Y1 posterior approximation with `Gaussbock`. The left figure depicts the matter density parameter (Ω_m) versus the Hubble constant (H_0), whereas the right figure shows the baryon density parameter (Ω_b) versus the scalar amplitude of density fluctuations (A_s). Contours for the importance-weighted samples generated with `Gaussbock` are drawn in blue, with contours for an `emcee` chain with 5.4 million samples across 54 walkers drawn in red. Darker and lighter shaded contour areas depict the 68% and 95% credible intervals, respectively. In addition to the same color coding as used in the contour plots, one-dimensional subplots for each parameter also show the unweighted distribution of `Gaussbock` samples in green, and the initial guess from which `Gaussbock` starts, obtained through a short-chained `emcee` run with 1000 steps per walker, in yellow. True means for DES Y1 data are indicated with dashed black lines to demonstrate the correct centering of both the fast approximation we employ in the experiment and the `Gaussbock` outputs.

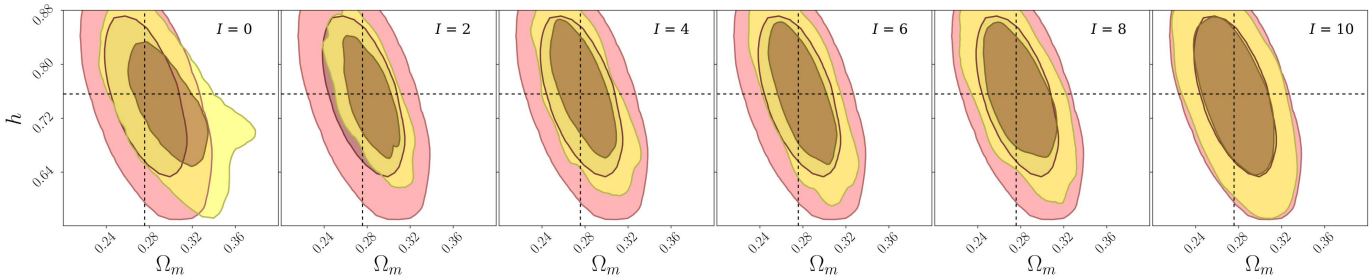

Figure 3. Gradual improvement of contours across `Gaussbock` iterations. The figure depicts, in yellow, the importance-weighted posterior approximations for the matter density parameter (Ω_m) versus the Hubble constant (H_0). Each panel indicates the respective number of iterations I in the upper right corner, for iteration numbers from the the set $\{0, 2, \ldots 10\}$ to cover easily visible morphing behavior before fine-tuning takes place. Contours for an `emcee` chain with 5.4 million samples across 54 walkers are drawn in red to serve as a target distribution and orientation point across panels. Darker and lighter shaded contour areas depict the 68% and 95% credible intervals, respectively. On the far left, at $I = 0$, the posterior approximation corresponds to the initial sample guess. True means for DES Y1 data are indicated with dashed black lines.

on an approximation of the full DES Y1 posterior, it offers a suitable testbed to prepare for the full-scale run described in Section 5.3.

Another factor of interest is the iterative behavior of our algorithm, as `Gaussbock` is supposed to continuously improve the agreement of its internally generated samples with the true posterior distribution. In Figure 3, we illustrate this behavior, showing the gradual improvement of the constraints. The plots depict the morphing and shifting behavior of `Gaussbock` samples for the number of iterations as even integers in the interval [0, 10]. The cosmological parameters chosen for this experiment are the same as in the left-hand panel of Figure 2.

The evolution across the different panels showcase the algorithm's ability to start from a very rough sample guess and gradually move toward the target distribu-

Table 2. Cosmological and nuisance parameter limits for a fast approximation of the DES Y1 posterior. The lower and upper limits shown as open intervals closely follow prior distribution features previously used by DES for data from the first year of observations (Abbott et al. 2018).

Category	Parameter	Interval
Cosmology	Ω_m	$[0.1, 0.9]$
	H_0	$[0.55, 0.9]$
	Ω_b	$[3 \cdot 10^{-2}, 7 \cdot 10^{-2}]$
	n_s	$[0.87, 1.07]$
	A_s	$[5 \cdot 10^{-10}, 5 \cdot 10^{-9}]$
	ω_ν	$[6 \cdot 10^{-4}, 10^{-2}]$
Lens galaxy bias	$b_1, \ldots, b_5$	$[0.8, 3.0]$
Shear calibration	$m_1, \ldots, m_4$	$[-0.1, 0.1]$
Intr. alignment	A_{IA}	$[-5.0, 5.0]$
	μ_{IA}	$[-5.0, 5.0]$
Source photo-z	$\Delta z_s^1, \ldots, \Delta z_s^4$	$[-0.1, 0.1]$
Lens photo-z	$\Delta z_l^1, \ldots, \Delta z_l^5$	$[-5 \cdot 10^{-2}, 5 \cdot 10^{-2}]$

tion. The latter is closely approximated by an extremely long `emcee` chain as an ideal sample. As demonstrated through this visualization, the algorithm first shifts generated samples toward the true mean with a lower-variance distribution, followed by incrementally spreading out to create a close fit to the target distribution.

5.2. *Exploration of scaling behavior*

In algorithms designed for the use with highly parallelized architectures, as well as in approaches for high-dimensional estimation problems, the question of how the algorithm in questions scales for different factors is important. For this reason, we now explore the scaling behavior of our algorithm. We quantify the time to convergence using the criterion introduced in Section 3, measured on the fast DES Y1 approximation covered in Section 5.1.

Higher-dimensional problems can, in general, be assumed to lead to a greater complexity of the estimation procedure, forcing `Gaussbock` to morph and shift the distribution in each iteration across more dimensions. We test our implementation for dimensionalities $3 \leq \dim(D) \leq 26$, up to the full set of cosmological and nuisance parameters in our DES Y1 approximation. We perform this parameter estimation 50 times for each number of dimensions to create confidence intervals, with the respective subset of parameters being randomly selected. In each case, we use the convergence threshold $0.01 \cdot \dim(D)$. The left panel of Figure 4 plots the number of iterations required to reach convergence versus the number of estimated parameters, showing the rise with problems of increased dimensionality.

The 95% confidence intervals around the average number of iterations to convergence highlight the larger variance with increasing numbers of parameters. The average number of 26.6 iterations for estimating the full set of 26 parameters provides an indicator for the full DES Y1 posterior computation in Section 5.3.

The second question in terms of scaling behavior targets the embarrassingly parallel part of our algorithm, as we can vary the number of samples drawn at each iteration. Although the ability to parallelize across large numbers of cores is one of the strengths of `Gaussbock`, and while access to parallel computing architectures is wide-spread in modern cosmology, the number of available cores for a given task still faces upper limits. As described in Section 4, a higher number of samples drawn from a given iteration's fitted model is generally preferable, which translates to a preference for a higher number of cores due to the subsequent parallelization of the truncated SIR step. This poses the question of the scaling behavior of this benefit, as the required number of iterations to convergence is expected to decrease with a higher number of samples per iteration.

The right panel of Figure 4 shows the scaling behavior of the required number of iterations to convergence versus the number of samples drawn from the fitted model during each iteration. We perform 50 `Gaussbock` runs per number of samples to create confidence intervals, in the interval [5000, 40000] and in steps of 5000.

Let I be the number of required iterations to convergence, C the number of available cores, and S the number of used samples per iteration. Then the total number of posterior value calculations per core over the course of a `Gaussbock` run is $I \cdot S \cdot C^{-1}$. Increasing numbers of samples constrain the variance of required iterations, and the dashed black line in the right panel of Figure 4 indicates an optimal trade-off (in terms of total core time) between the two variables as $\min(I \cdot S)$ at $S = 15000$ for the number of samples, which informs our input choices in Section 5.1. This visualization also bears resemblance to the 'elbow criterion' in cluster analysis, which determines the optimal number of clusters by plotting that number against the explained variance (Thorndike 1953).

Lastly, we investigate the convergence behavior of `Gaussbock` as a follow-up to Figure 3, to ensure that both the convergence check itself and the recommended calculation of a convergence threshold behave as intended. The algorithm is run on the same parameter estimation problem as in Section 5.1, for a total of 27 iterations to cover the previously computed mean number of iterations to convergence of 26.6. As for previous tests, we run this experiment 50 separate times

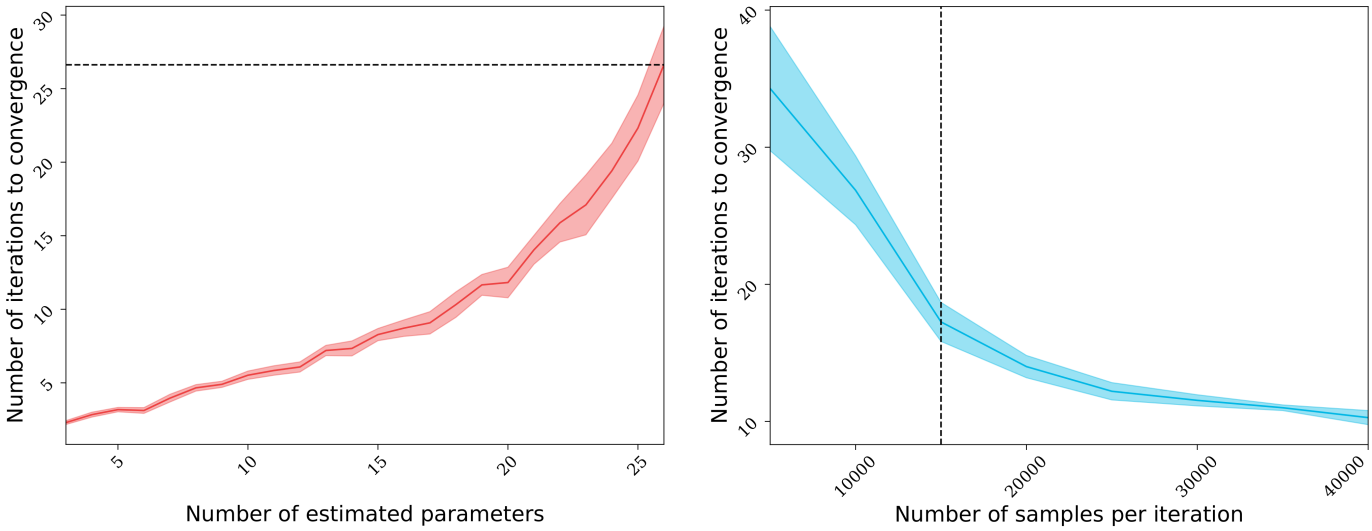

Figure 4. Relationship between time to convergence, dimensionality, and the number of samples per iteration for `Gaussbock`. The left panel shows the number of iterations needed to achieve convergence, as a function of the dimensionality of the problem. The dashed black line indicates the mean number of iterations (26.6) needed for the full 26D DES Y1 parameter set. The right panel shows the number of iterations before convergence, as a function of the number of importance samples taken at each iteration, in steps of 5000. The dashed line marks the 'elbow criterion' for the trade-off in terms of time requirements from iterations and sample size, at 15000 samples. In both panels, the central line shows the mean and the shaded band the 95% confidence intervals over 50 simulations per point.

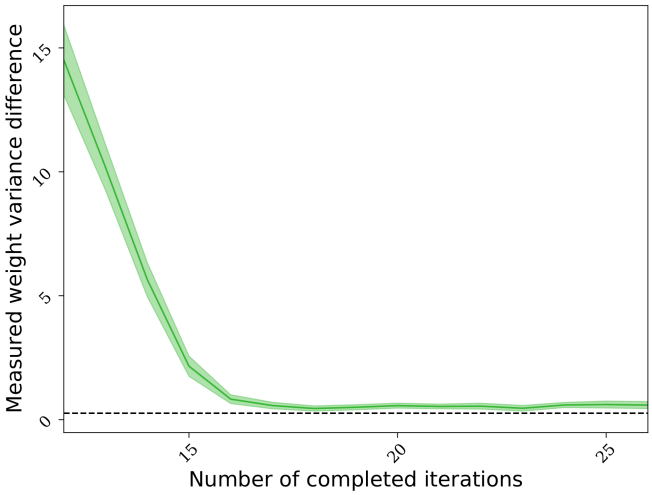

Figure 5. Convergence behavior of `Gaussbock` for the number of completed iterations in approximated 26D DES Y1 analyses. The figure shows the inter-iteration change in variances of the logarithmic weights, used as a convergence criterion, with the dashed line marking the default convergence threshold for this problem. The mean value over 50 runs is shown as the central line, and the shaded band shows the 95% confidence interval.

to generate 95% confidence intervals. The results are shown in Figure 5, starting after the first 10 iterations to cover fine-tuning behavior after the initial morphing and shifting explored in Figure 3, and with the dashed black line indicating the convergence threshold set to $0.01 \cdot \dim(D) = 0.26$. The figure, showing a remarkably consistent and well-constrained behavior, demonstrates both convergence behavior for the threshold calculation and narrow confidence intervals for multiple experiments.

5.3. *The full Dark Energy Survey posterior*

In order to expose our method to a fully realistic experiment without approximations, we apply `Gaussbock` to the full DES posterior from the DES Y1 experiments and data release (Abbott et al. 2018; Abbott et al. 2018; Krause et al. 2017). We use the public `CosmoSIS` implementation of the public Y1 likelihood, which includes `CAMB` as descibed by Lewis & Bridle (2002) and Howlett et al. (2012), and `Halofit` as introduced in Smith et al. (2003) and Takahashi et al. (2012) to compute distances and matter power spectra, `CosmoSIS`-specific modules for the Limber integral and other intermediate steps, and `Nicaea`[4] for the computation of real-space correlations from Fourier space values (Kilbinger et al. 2009). Since the public implementation of the Y1 likelihood differs very slightly from the released chains, we rerun the model referred to as `d_l3` in the public DES Y1 chains using `MultiNest` for an identical comparison. The experiment starts with the same initial sample guess via a short-chained `emcee` run that we use for our fast approximation of the DES Y1 posterior in Section 5.1, demonstrating that our ap-

[4] http://www.cosmostat.org/software/nicaea

proach is able to start from approximative guesses that only partially fall within the vicinity of the target posterior and are not necessarily based on calculations using the actual target in question.

Making use of `Gaussbock`'s innate embarrassing parallelism, we run this experiment on supercomputing facilities of the National Energy and Scientific Research Computing Center (NERSC)[5] (He et al. 2018). We run on 32 nodes of the Cori computer, for a total of 1024 cores and 2048 threads. The results below were generated in approximately two hours in this configuration, showcasing the total runtime advantage of our approach. With the runtime scaling being inversely linear with the number of cores, up to the number of samples used per iteration due to the model-fitting process not requiring a lot of time, up to 15000 cores can be used in an idealized scenario for our experimental setup to gain a further order-of-magnitude reduction. In order to make use of existing posterior implementations, we employ `CosmoSIS` to use `Gaussbock` with the DES Y1 posterior (Zuntz et al. 2015).

Table 3 lists the cosmological parameters as estimated by both `Gaussbock` and its comparison baseline, meaning the fiducial `MultiNest` run, demonstrating a satisfactory level of agreement for both means and credible intervals. In addition to the cosmological parameters shown in the experiments for Figures 2 and 6, the table also includes the scalar spectral index n_s and the massive neutrino density ω_ν, covering the full set of cosmological parameters previously listed in Table 2.

Table 3. Cosmological parameters for DES Y1 data. The table shows figures of merit for common cosmological parameters used in the original DES Y1 experiments, with the latter's implementation of `MultiNest` and, for comparison, the results for a highly parallel `Gaussbock` run.

Parameter	`MultiNest`	`Gaussbock`
Ω_m	$0.276^{+0.031}_{-0.031}$	$0.275^{+0.029}_{-0.026}$
H_0	$0.787^{+0.080}_{-0.106}$	$0.781^{+0.078}_{-0.080}$
Ω_b	$0.056^{+0.010}_{-0.012}$	$0.057^{+0.009}_{-0.013}$
n_s	$1.020^{+0.043}_{-0.064}$	$1.013^{+0.043}_{-0.065}$
A_s	$2.470^{+0.510}_{-0.440} \times 10^{-9}$	$2.430^{+0.420}_{-0.400} \times 10^{-9}$
ω_ν	$5.100^{+2.900}_{-2.800} \times 10^{-3}$	$5.000^{+3.000}_{-2.800} \times 10^{-3}$

Figure 6 shows the posterior contours for both the `d_l3` rerun with `MultiNest` and the `Gaussbock` result in red and blue, respectively. Both matter and baryon density parameters, Ω_m and Ω_b, are shown to match the baseline computation well, whereas the Hubble parameter H_0 and scalar amplitude of density fluctuations A_s are in reasonable agreement, but do not correctly recover the tails of the posterior distribution. An exploration of the 26-dimensional approximation shows that `Gaussbock` accurately models the parameters which are well-constrained, but fails to recover the tails on unconstrained parameters like H_0 and A_a that have very broad intervals, as listed in Table 2. Where possible, it might help to provide narrower constraints for such parameters. In addition, Figure 6 shows the joint posterior of the two intrinsic alignment parameters, A_{IA} and μ_{IA} in the right panel.

The results demonstrate the ability of `Gaussbock` to recover non-Gaussian shapes of correlated parameters to a high degree of accuracy, as can be seen in the 2D posterior shapes for the fiducial `MultiNest` and `Gaussbock` runs, as well as in the agreement between histograms in the figure. The effective sample size for this `Gaussbock` run is $N_{\mathrm{eff}} = 2104$, compared to $N_{\mathrm{eff}} = 4316$ for the original `MultiNest` chain, although with a smaller overall runtime for our algorithm.

While the results are not in near-perfect agreement, as is the case for the fast truncated Gaussian approximation in Section 5.1, a trade-off between considerably reduced runtime and accuracy is to be expected analogous to the No Free Lunch Theorem in optimization (Wolpert & Macready 1997). The described experiment on the full DES Y1 posterior makes use of `Gaussbock`'s adaptive default behavior and, for the number of samples per iteration, is based on our fast approximation, so fine-tuning to a specific application case can be expected to further improve the performance of the algorithm. Other reasons for the results not showing the same goodness of fit for all parameters, as observed in Section 5.1, are a diminished smoothness of posteriors and less Gaussian tails, which we discuss in Section 6.

5.4. *Stress tests on additional distributions*

In this subsection we run `Gaussbock` on distributions with more challenging features to determine when it starts to fail. As outlined in Section 4, KDE is a powerful density estimation technique, but faces issues in higher-dimensional problems (O'Brien et al. 2016). In this experiment, we exemplify the built-in default to use KDE for problems in which $\dim(D) \leq 2$, allowing `Gaussbock` to make use of the method most suitable to a given problem. For this purpose, we construct a posterior of three approximately equilateral triangles with a flat posterior surface, meaning that posterior values

[5] https://www.nersc.gov

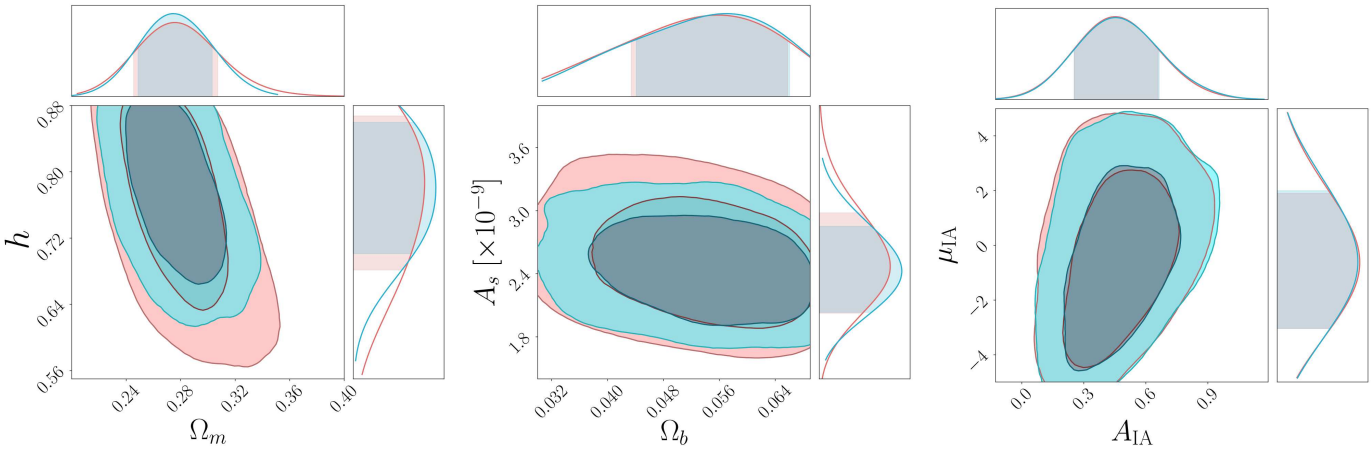

Figure 6. DES Y1 posteriors with `Gaussbock`. The left panel depicts the matter density parameter (Ω_m) versus the Hubble constant (H_0), the middle figure shows the baryon density parameter (Ω_b) versus the scalar amplitude of density fluctuations (A_s), and the right figure shows the two intrinsic alignment parameters ($A_{\rm IA}$, $\mu_{\rm IA}$). Contours for the importance-weighted samples generated with `Gaussbock` are drawn in blue, with contours for the original nested sampling implementation as used by DES drawn in red. Darker and lighter shaded contour areas depict the 68% and 95% credible intervals, respectively, with the same levels shaded in the histograms.

are uniform across the triangle shapes. Due to the convergence criterion of `Gaussbock`, which we discuss in Section 3, being geared toward the use of a VBGMM as its primary application in high-dimensional setting, we set the number of iterations to 20. We let the initial sample guess be generated automatically with the same number as for previous experiments in Section 5.1, and let `Gaussbock` use its default behavior for optional inputs.

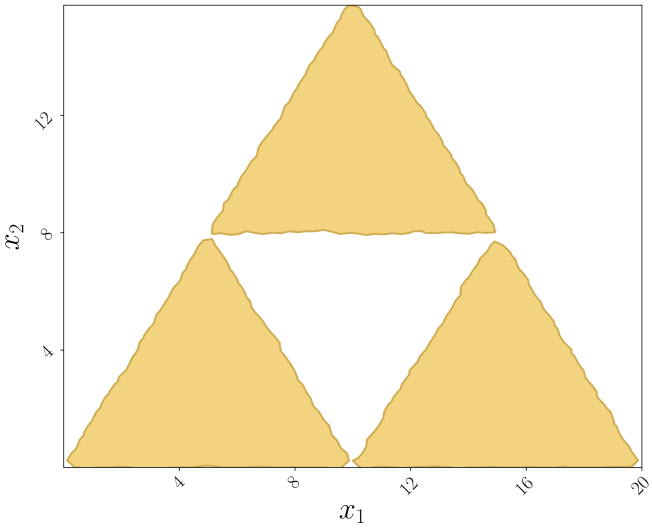

Figure 7. Approximation of a hard-to-estimate posterior with `Gaussbock`. The two-dimensional posterior distribution features uniform values across the surface of three triangles. With a completely flat distribution of the posterior shape, the importance-weighted sample contours in the plot show the 95% credible interval for the generated samples.

The results of this low-dimensional parameter estimation experiment is shown in Figure 7, with 95% credible intervals for the flat-surface posterior demonstrating the ability of `Gaussbock` to approximate complex shapes with pronounced edges and corners. The three separate triangles are clearly reconstructed through the importance-weighted samples generated by the algorithm, validating its integrated KDE functionality for low-dimensional estimation problems.

Next, we consider similar stress tests based on those described in Hobson & Feroz (2008) and Feroz et al. (2009). First, we test with a posterior in the form of a double Gaussian shell, as described in Allanach & Lester (2008),

$$\mathcal{L}(\theta) = C(\theta; \mathbf{c_1}, w, r) + C(\theta; \mathbf{c_2}, w, r), \tag{10}$$

where

$$C(\theta; \mathbf{c}, w, r) = \mathcal{N}(|\theta - \mathbf{c}|\,; r, w^2). \tag{11}$$

At low dimensions, Gaussbock can sample effectively from such a distribution; the results from a 2D example with $w = 0.1$ and $r = 2$ are shown in Figure 8. The samples correctly trace the distribution, with a close-to-ideal match between the brute-force percentiles and the fraction of samples inside them. At moderate dimensions, from around 5D, Gaussbock fails on the sharp edges in this distribution, as the required number of Gaussians to capture the full shape becomes too high.

Next, we consider sharp edges that are poorly fit by Gaussian mixtures as another possible failure cases. We sample from

$$\mathcal{L} = \begin{cases} \exp -2 \cdot |\mathbf{x}|, & \text{if } \forall x \in \mathbf{x} : x > 0 \\ 0, & \text{else} \end{cases} \tag{12}$$

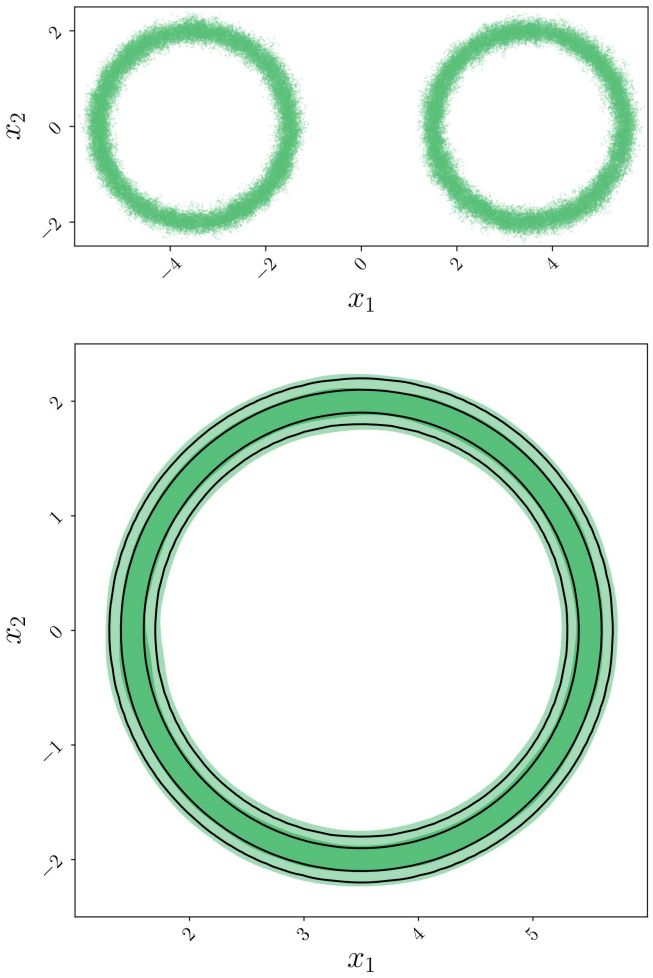

Figure 8. Samples from a 2D Gaussian shell distribution. The upper panel shows a scatter plots of the resulting `Gaussbock` samples, while the lower panel zooms in on one of the two shells. For the latter, we show inner 68% and outer 95% contours from a brute-force grid evaluation in black, and KDE on `Gaussbock` samples as blue-shaded regions, with darker and lighter shaded contour areas depicting the 68% and 95% credible intervals, respectively. At higher dimensions, `Gaussbock` fails on such distributions.

using a 4D example. This form has a sharp edge at $\mathbf{x}_i = 0$ in each dimension. Figure 9 shows the 1D distribution of one of the four parameters, as sampled using `Gaussbock`, `emcee`, and a brute-force evaluation. Both samplers undersample at this boundary[6], and this effect will worsen for `Gaussbock` at higher dimension.

Finally, as a multimodal example, we consider sets of identical Gaussians, with centers arranged in a Latin

[6] Many sampling methods based on Markov chains can suffer from repulsive effects at sharp edges of distributions, since proposals to points near the boundary can only happen from one direction; a variety of methods have been used to correct for this behavior (Ahmadian et al. 2011).

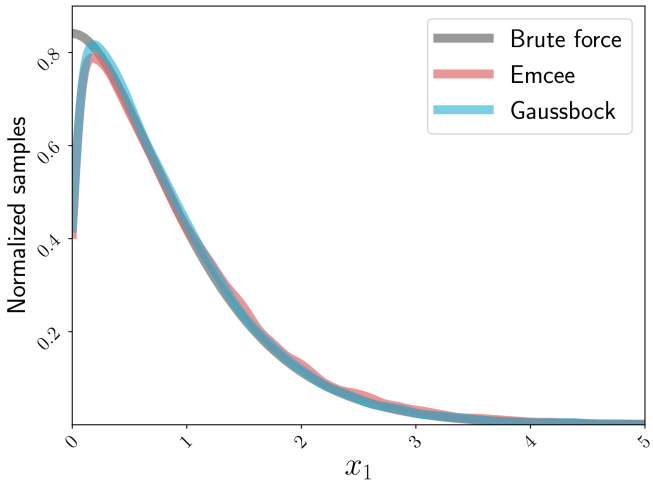

Figure 9. Sampling behavior of `Gaussbock` on the distribution in Equation 12, with a sharp boundary in 4D, compared to a long-chained `emcee` run and a brute-force evaluation. Both samplers underestimate the PDF near the edge, although `Gaussbock` maintains a slightly smoother adherence to the true distribution otherwise.

hypercube formation so that they do not overlap in any dimension. The algorithm, starting from a random scattering throughout the space, finds all the modes for dimensions up to about six, as shown in Figure 10. At higher dimensions, the algorithm often misses some of the modes; this is an important failure case that is based in the reliance on an initial sample provided by either the built-in affine-invariant MCMC sampler or a method of choice. If the latter fails to catch at least part of some modes, the algorithm is unlikely to recover them.

It should be noted that most distributions found in the additional tests of this section are not usually found in the intended field of application, cosmological parameter estimation, but serve as a demonstration of the method's capabilities for classical tests found in the statistical literature, and could be of use in other application areas.

6. DISCUSSION

The primary advantage of our approach is the considerable reduction in the required runtime, given a large-enough number of cores available for parallelization. This strength offers a way to tackle increasing complexities in cosmological parameter estimation for current and upcoming surveys such as LSST and Euclid (Amendola et al. 2018). Since cosmological parameter estimation efforts rely on computationally costly posterior evaluations, the embarrassing parallelization of their calculation allows for an immense speed-up in comparison to standard MCMC approaches. This reduction in total runtime comes, however, at the cost of an increase in the required core time, meaning the number of com-

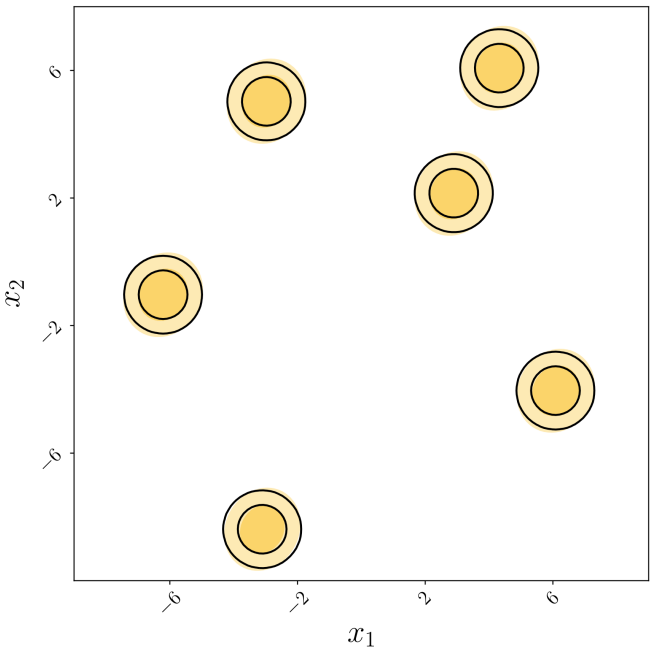

Figure 10. A 2D projection of a six-dimensional distribution with six Latin hypercube-located Gaussian modes. We show a KDE on `Gaussbock` samples as yellow-shaded regions, with darker and lighter shaded contour areas depicting the 68% and 95% credible intervals, respectively. The algorithm typically finds all the modes up to about 6D, and then begins to miss them at higher dimensions due to the difficulty of catching them in the initial sample generation..

puting hours necessary to achieve suitable results. For this reason, and assuming a sufficiently costly posterior evaluation, making use of `Gaussbock`'s parallelization capabilities is a requirement rather than an optional feature, as demonstrated in Section 5.3.

A direct comparison to MCMC methods is a double-edged sword in that such methods, run for a very large number of steps, provide a close fit to the true posterior. The downside of MCMC approaches is that they tend to not scale well with the number of dimensions, and that they are only parallelizable over the number of walkers. This means that computationally expensive likelihoods provide an obstacle to implementations such as `emcee` (Foreman-Mackey et al. 2013). While nested sampling methods circumvent this restriction by requiring fewer posterior evaluations, they rely on assumptions about perfect and independent samples and can sometimes underestimate an asymptotically Gaussian sampling error. In many cases, though, they can be highly effective, for both posterior and evidence estimation, depending on the problem at hand (Chopin & Robert 2010).

As mentioned in Section 5.3, posteriors based on real-world survey data may have a less smooth posterior surface, which can hamper the effectiveness of the truncated SIR step used in our approach. Adjusting the 'truncation_alpha' input can alleviate this issue for isolated samples with higher posterior values, although a more effective solution is to increase the number of samples drawn from the posterior approximation of a given iteration of the algorithm. This approach does, in turn, require either a correspondingly larger number of cores or additional runtime. Alternatively, the initial sample guess to which the first-iteration model is fitted can be based on a longer-chain `emcee` run. As a result, this approach offers a better approximation of the posterior to start from, as it more closely resembles the target distribution and leads to broader coverage of relevant areas. We hope that the presented work will lead to further investigations of this and related parallelized iterative approaches to parameter estimation, alleviating the issues arising from increased computational demands in inference based on modern surveys.

Apart from cases with sufficiently smooth posteriors and well-constrained parameters, `Gaussbock` also offers a way to quickly approximate a posterior to reasonable degrees. For this purpose, we recommend using either uniform-random samples from an n-sphere scaled to the admissible ranges or, if feasible, samples from a better-suited distribution like an n-dimensional Gaussian to provide an initial sample guess covering the posterior area. The reason for such approaches is the elimination of the need for computationally more expensive sample guess generators such as short-chained `emcee` runs, which require costly evaluations of the posterior. While short chains are fast in comparison to exhaustive runs of MCMC methods, runtimes should be kept to a minimum for fast approximations in order to provide an edge in speed over alternative approaches.

An additional use case pertains to lower-dimensional problems, or scenarios with posterior evaluations that are sufficiently cheap to compute, and offers a way to achieve very tight fits to posteriors that are hard to approximate and feature clean cuts, with an example given in Section 5.4 and one commonly-encountered example of such posterior shapes being truncated Gaussians. The suitability for the latter type also extends to higher dimensions, as we demonstrate with the truncated Gaussian approximation of the 26-dimensional DES Y1 posterior in Section 5.1. For the latter, as described in Section 5.3, an important finding is that `Gaussbock` accurately models well-constrained parameters, but can have trouble to recover the tails on unconstrained parameters perfectly. For that reason, setting sensible parameter constraints as one of the three required inputs to the implementation is strongly advised.

Unlike in most MCMC methods, the final mixture model is an optional output of our implementation, which can be saved and used again at a later point. It can act as an approximate but analytic description of the posterior, allowing, for example, the subsequent drawing of an arbitrary number of samples for which importance weights can be calculated and which can be easily disseminated. In this context, our approach offers a way to easily exchange and compare posterior approximations based on different datasets, with mixture models whose components can be combined.

For common problems faced by contemporary research in cosmology, `Gaussbock` offers a considerable speed-up. This is especially relevant for upcoming missions with larger numbers of parameters, for which our approach provides a way to quickly compute high-fidelity posterior approximations and the underlying mixture model. While, in this work, we use a wrapper to run `Gaussbock` through `CosmoSIS` on NERSC facilities, a complete integration into `CosmoSIS` will further enhance the ease of access to our methodology. Regarding the scaling behavior tested in Section 5.2, a higher number of dimensions leads to a higher number of iterations to reach convergence, as demonstrated in Figure 4. `Gaussbock` also benefits from an as-close-as-possible fit to the true posterior for the initial sample to start from. In cases in which such a sample guess is available, it lends an advantage to the method's performance when compared to using the built-in affine-invariant MCMC sampler. Notably, the ability to feed an arbitrary set of initial samples into the tool also means that `Gaussbock` can be combined with any sampling algorithm to create such an initial sample, allowing users to employ cutting-edge methods of their choice to make full use of the current statistical literature and personal preferences.

In terms of its internal functionality, our approach inherently lends itself to combating issues with defaulting cores, as the failure or a subset of processes to return importance values can be safely ignored. The respective parameter sets can simply be omitted from the set of samples used to approximate the posterior in a given iteration, using the large-enough amount of remaining parameter sets to fit the model in a given iteration. While the capability to do so is not part of our implementation and is primarily of interest for large-scale cloud computing, our code easily lends itself to being extended toward this safety redundancy.

7. CONCLUSION

In this paper, we introduce and apply `Gaussbock`, a novel approach to cosmological parameter estimation that makes use of recent advances in machine learning and statistics. By coupling variational Bayesian GMMs with a truncation-based extension of importance sampling in an iterative approach with a convergence criterion, our method offers an embarrassingly parallel way to achieve high-speed parameter estimation for problems with computationally expensive likelihood calculations.

We initially test `Gaussbock` on a fast approximation of the DES Y1 posterior to demonstrate its capabilities on a high-dimensional realistic example, and to investigate scaling relations and the effectiveness of the convergence criterion, both of which prove to be well-behaved. We then apply our method to the full DES Y1 posterior, making use of `Gaussbock`'s built-in MPI capabilties to run it on NERSC supercomputing facilities. The results showcase the immense speed-up that constitutes the primary strength of our method, achieving a good fit to the original DES approach of using `MultiNest`.

While achieving excellent fits in most cases across our experiments, we observe that less Gaussian posteriors of unconstrained parameters result in a slightly worse fit to the tails of the distribution and discuss the potential issues arising from less smooth posterior surfaces. In addition, we stress-test the algorithm using more complex distributions. We also demonstrate that `Gaussbock` achieves tight fits to hard-to-approximate posteriors such as double Gaussian shells, scattered multivariate Gaussians, and exponential distributions in lower dimensions. The reliance on an initial sample guess roughly covering the areas of interest, however, means that it will break down if the latter is not the case, for example if modes of a multivariate distribution are not caught in that initial sample. In addition, we verify that our method, like other parameter estimation techniques based on Gaussian mixture models, is limited by the degree to which distributions can be formalized as a weighted mixture of Gaussians, which becomes problematic if, for example, facing Gaussian shells of moderate to high dimensionality.

We implement `Gaussbock` as a pure-Python package to conduct our experiments described in this paper. In doing so, we also provide the astronomy community with a user-friendly and readily installable implementation of `Gaussbock`, bearing the same name. While our method is developed specifically with contemporary parameter estimation problems in cosmology in mind, it represents a general-purpose inference tool applicable to many problems dealing with high-dimensional parameter estimation with computationally costly posteriors. As a result, our work contributes to the wider field of estimation theory in addition to current and upcoming astronomical surveys.

ACKNOWLEDGMENTS

We thank members of the Dark Energy Survey collaboration for making DES data publicly available. BM acknowledges the support of a Principal's Career Development Scholarship from the University of Edinburgh. JZ acknowledges a Chancellor's Fellowship, also from the University of Edinburgh.

REFERENCES

Abbott, T., Abdalla, F. B., Aleksić, J., et al. 2016a, MNRAS, 460, 1270

Abbott, T., Abdalla, F. B., Allam, S., et al. 2016b, PhRvD, 94, 022001

Abbott, T. M. C., Abdalla, F. B., Alarcon, A., et al. 2018, PhRvD, 98, 043526

Abbott, T. M. C., Abdalla, F. B., Allam, S., et al. 2018, ApJS, 239, 18

Aghanim, N., Akrami, Y., Ashdown, M., et al. 2018, arXiv e-prints, arXiv:1807.06209

Ahmadian, Y., Pillow, J. W., & Paninski, L. 2011, Neural Comput., 23, 46

Aitken, S., & Akman, O. E. 2013, BMC Syst. Biol., 7, 72

Akeret, J., Refregier, A., Amara, A., Seehars, S., & Hasner, C. 2015, JCAP, 2015, 043

Akeret, J., Seehars, S., Amara, A., Refregier, A., & Csillaghy, A. 2013, Astron. Comput., 2, 27

Allanach, B., & Lester, C. 2008, Computer Physics Communications, 179, 256

Allison, R., & Dunkley, J. 2014, MNRAS, 437, 3918

Alsing, J., Wandelt, B., & Feeney, S. 2018, MNRAS, 477, 2874

Amendola, L., Appleby, S., Avgoustidis, A., et al. 2018, Living Rev. Relativ., 21, 2

Antoniak, C. E. 1974, Ann. Statist., 2, 1152

Bardenet, R., Doucet, A., & Holmes, C. 2014, in Proceedings of the 31st International Conference on Machine Learning (ICML 2014), 405–413

Bernardo, J., & Smith, A. F. M. 1994, Bayesian theory (Hoboken: John Wiley & Sons, Inc.)

Betoule, M., Kessler, R., Guy, J., et al. 2014, A&A, 568, A22

Bishop, C. M. 2006, Pattern recognition and machine learning (Heidelberg, Germany: Springer-Verlag)

Blei, D. M., & Jordan, M. I. 2006, Bayesian Anal., 1, 121

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. 2017, J. Am. Stat. Assoc., 112, 859

Bonassi, F. V., Lingchong, Y., & West, M. 2011, Stat. Appl. Genet. Mol. Biol., 10, 49

Brooks, S., Gelman, A., Jones, G., & Xiao-Li, m. 2011, Handbook of Markov chain Monte Carlo (New York: Chapman & Hall (CRC Press))

Cappé, O., Guillin, A., Marin, J.-M., & Robert, C. P. 2004, J. Comput. Graph. Stat., 13, 907

Chopin, N., & Robert, C. 2010, Biometrika, 97, 741

Christensen, N., & Meyer, R. 1998, PhRvD, 58, 082001

Christensen, N., Meyer, R., Knox, L., & Luey, B. 2001, Class. Quantum Grav., 18, 2677

Del Pozzo, W. 2012, PhRvD, 86, 043011

Del Pozzo, W., Li, T. G. F., & Messenger, C. 2017, PhRvD, 95, 043502

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. 1987, Phys. Lett. B, 195, 216

Fan, Y., Nott, D. J., & Sisson, S. A. 2013, Stat, 2, 34

Ferguson, T. S. 1973, Ann. Statist., 1, 209

Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, PASP, 125, 306

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. 2013, Bayesian data analysis (New York: Chapman & Hall (CRC Press))

Geman, S., & Geman, D. 1984, IEEE Trans. Pattern Anal. Mach. Intell., 6, 721

Gershman, S. J., & Blei, D. M. 2012, J. Math. Psychol., 56, 1

Goodman, J., & Weare, J. 2010, Commun. Appl. Math. Comput. Sci., 5, 65

Hajian, A. 2007, PhRvD, 75, 083525

Handley, W. J., Hobson, M. P., & Lasenby, A. N. 2015, MNRAS, 453, 4384

Hastings, W. K. 1970, Biometrika, 57, 97

He, Y., Cook, B., Deslippe, J., et al. 2018, Concur. and Computat.: Practice and Experience, 30, e4291

Higson, E., Handley, W., Hobson, M., & Lasenby, A. 2018, Bayesian Anal., 13, 873

Hjort, N. L., Holmes, C., Mueller, P., & Walker, S. G. 2010, Bayesian nonparametrics: Principles and practice (Cambridge, UK: Cambridge University Press)

Hobson, M. P., & Feroz, F. 2008, MNRAS, 384, 449

Hobson, M. P., Jaffe, A. H., Liddle, A. R., Mukherjee, P., & Parkinson, D. 2009, Bayesian methods in cosmology (Cambridge, UK: Cambridge University Press)

Hoffmann, M. D., & Gelman, A. 2014, J. Mach. Learn. Res., 15, 1593

Howlett, C., Lewis, A., Hall, A., & Challinor, A. 2012, JCAP, 2012, 027
Ionides, E. L. 2008, J. Comput. Graph. Stat., 17, 295
Ishida, E. E. O., Vitenti, S. D. P., Penna-Lima, M., et al. 2015, Astron. Comput., 13, 1
Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. 1999, Mach. Learn., 37, 183
Kacprzak, T., Kirk, D., Friedrich, O., et al. 2016, MNRAS, 463, 3653
Kahn, H., & Marshall, A. W. 1953, Oper. Res., 1, 263
Keeton, C. R. 2011, MNRAS, 414, 1418
Kilbinger, M., Benabed, K., Guy, J., et al. 2009, A&A, 497, 677
Kilbinger, M., Wraith, D., Robert, C. P., et al. 2010, MNRAS, 405, 2381
Knox, L., Christensen, N., & Skordis, C. 2001, ApJ, 563, L95
Krause, E., & Eifler, T. 2017, MNRAS, 470, 2100
Krause, E., Eifler, T. F., Zuntz, J., et al. 2017, arXiv e-prints, arXiv:1706.09359
Lewis, A., & Bridle, S. 2002, PhRvD, 66, 103511
Liddle, A., Parkinson, D., & Mukherjee, P. 2006, Astron. Geophys., 47, 4.30
MacKay, D. J. C. 2003, Information theory, inference and learning algorithms (Cambridge, UK: Cambridge University Press)
Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. 1953, JChPh, 21, 1087
Mukherjee, P., Parkinson, D., & Liddle, A. R. 2006, ApJ, 638, L51
Murphy, K. P. 2012, Machine learning: A probabilistic perspective (Cambridge, USA: The MIT Press)
Neiswanger, W., Wang, C., & Xing, E. P. 2014, in Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI'14), 623–632
O'Brien, T. A., Kashinath, K., Cavanaugh, N. R., Collins, W. D., & O'Brien, J. P. 2016, Comput. Stat. Data Anal., 101, 148
Papamakarios, G., & Murray, I. 2016, in Advances in Neural Information Processing Systems 29, ed. D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Red Hook, USA: Curran Associates), 1028–1036
Peterson, C., & Anderson, J. R. 1987, Complex Syst., 1, 995
Peterson, C., & Hartman, E. 1989, Neural Netw., 2, 475
Price-Whelan, A. M., & Foreman-Mackey, D. 2017, J. Open Source Softw., 2, 357
Robert, C., & Casella, G. 2011, Statistical Science, 26, 102
Robert, C. P., & Casella, G. 2004, Monte Carlo statistical methods (Heidelberg, Germany: Springer-Verlag)
Robert, C. P., Elvira, V., Tawn, N., & Wu, C. 2018, WIREs Comput. Stat., 10, e1435
Saha, P., & Williams, T. B. 1994, AJ, 107, 1295
Sethuraman, J. 1994, Stat. Sin., 4, 639
Skilling, J. 2006, Bayesian Anal., 1, 833
Smith, R. E., Peacock, J. A., Jenkins, A., et al. 2003, MNRAS, 341, 1311
Stumpf, M. P. H., Kirk, P., & Johnson, R. 2014, Bioinformatics, 31, 604
Takahashi, R., Sato, M., Nishimichi, T., Taruya, A., & Oguri, M. 2012, ApJ, 761, 152
Thorndike, R. L. 1953, Psychometrika, 18, 267
Torrie, G. M., & Valleau, J. P. 1977, J. Comput. Phys., 23, 187
Trotta, R. 2008, Contemp. Phys., 49, 71
Trotta, R., Jóhannesson, G., Moskalenko, I. V., et al. 2011, ApJ, 729, 106
Verde, L., Feeney, S. M., Mortlock, D. J., & Peiris, H. V. 2013, JCAP, 2013, 013
Wang, S., Wang, Y.-F., & Xia, D.-M. 2018, Chin. Phys. C, 42, 065103
Wilkinson, D. J. 2005, in Handbook of parallel computing and statistics (New York: Chapman & Hall (CRC Press)), 477–513
Wolpert, D. H., & Macready, W. G. 1997, IEEE Trans. Evol. Comput., 1, 67
Wraith, D., Kilbinger, M., Benabed, K., et al. 2009, PhRvD, 80, 023507
Zuntz, J., Paterno, M., Jennings, E., et al. 2015, Astron. Comput., 12, 45